Exploring Accelerator Integration With Core-V eXtention InterFace (CV-X-IF) for Kyber

Alessandra Dolmeta [1,2], Stefano Di Matteo [2,3], Emanuele Valea[3]

1 Politecnico di Torino, DET - Dipartimento di Elettronica e Telecomunicazioni , Turin, Italy, 2 Université Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France, 3 Université Grenoble Alpes, CEA, List, F-38000 Grenoble, France

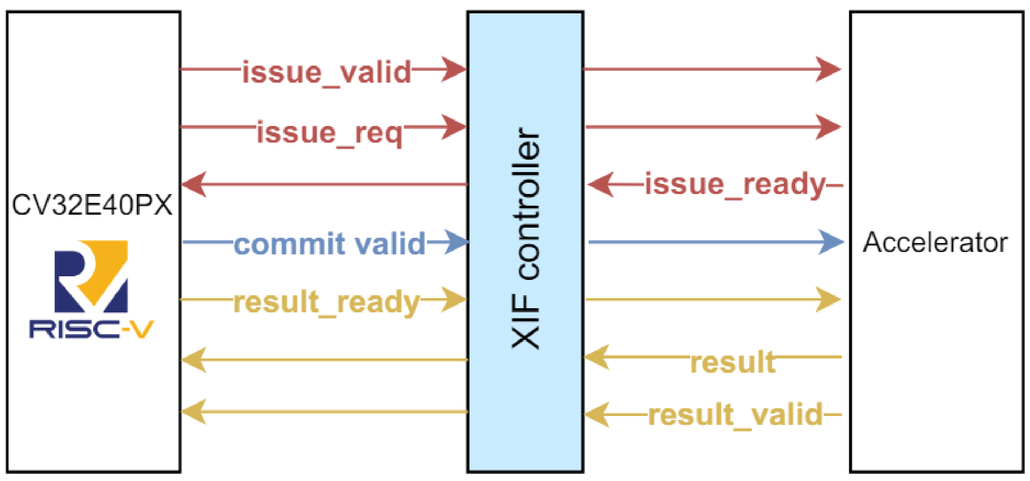
Introduction

In this era of advancing technology, the progression of embedded systems has experienced a significant shift towards incorporating specialized accelerators into microcontrollers, intending to boost performance and efficiency. **RISC-V** fosters processor innovation and enables open-source hardware development with the integration of domain-specific ISA extensions, accelerators, and co-processors.

A crucial aspect of this integration is the degree of connectivity between the central processing unit and these accelerators. **Loosely-coupled accelerators**, typically memory-mapped and connected via system buses, handle compute-heavy tasks with large data sets. On the other hand, **tightly-coupled accelerators** reside within the CPU's microarchitecture, necessitating core and toolchain modifications.

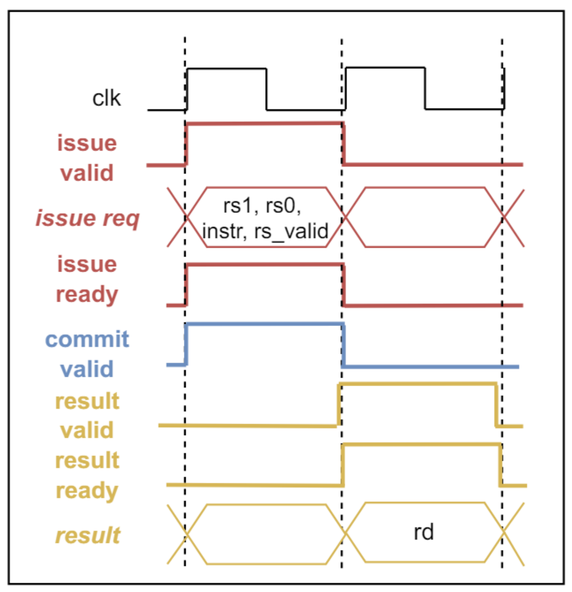
Proposed CV-X-IF Integration

The emerging **eXtension Interface** (CV-X-IF) [**1**] allows a seamless support of co-processors. In fact, the CV-X- IF can enhance the CPU with custom or standardized instructions for co-processor support, still without altering the CPU's decode unit.



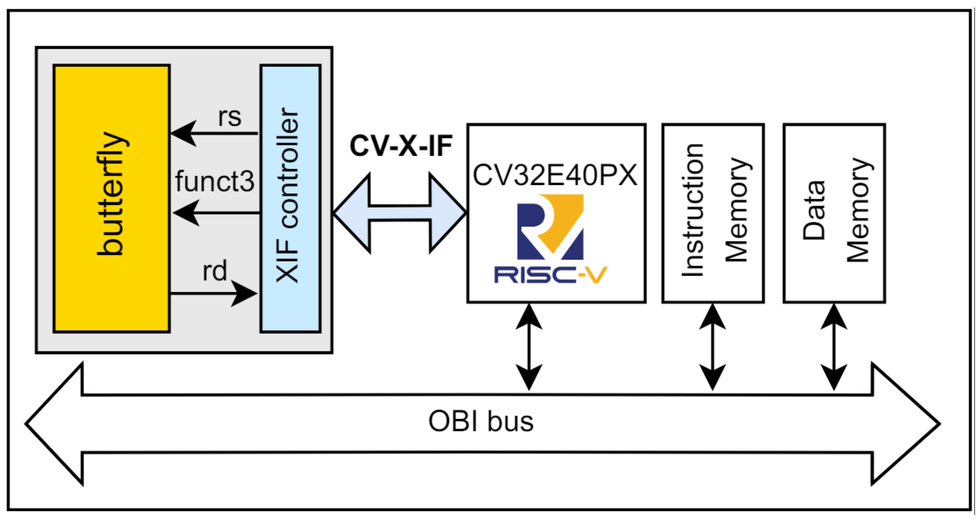
The **XIF controller:**

- ❑ scrutinizes instruction opcodes from instruction field of **issue req**.
- ❑ activates the signal **issue ready** if it recognizes a match.
- ❑ validates input-source registers (**rs valid** signals, **issue_req** package).
- ❑ executes instructions.
- ❑ if the core is ready, it is written back in the register file through the **result interface**.

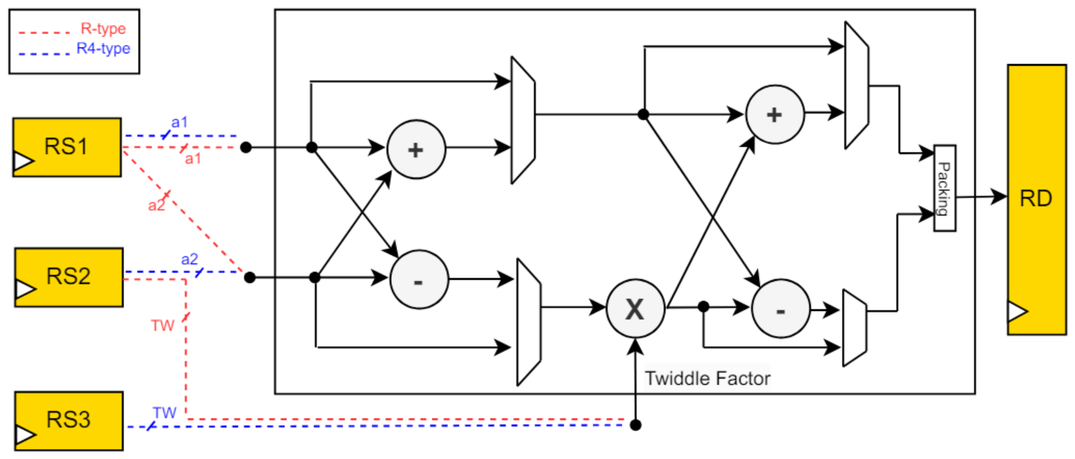


Implementation

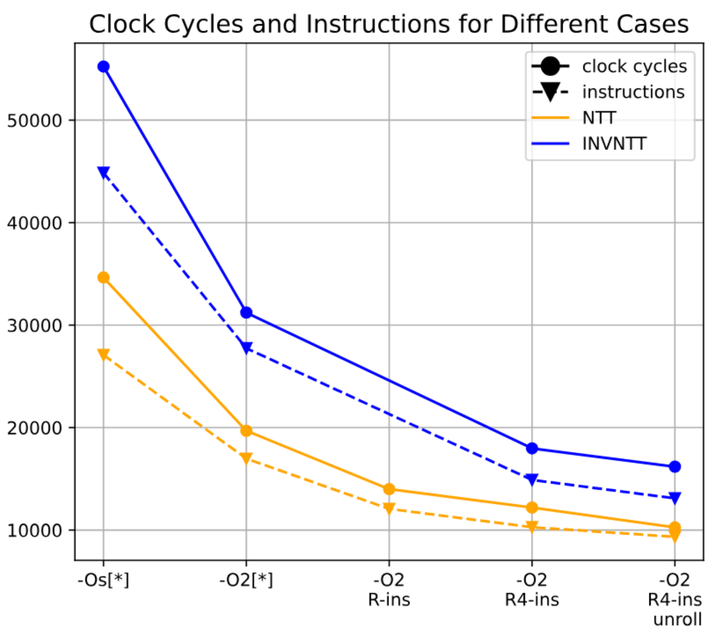
We propose the butterfly unit reported in this work [3], and we connected it to the RISC-V core through the CV-X-IF interface. **Number Theoretic Transform** (NTT) is a mathematical operation that allows to efficiently compute polynomial multiplications; it is used in lattice-based cryptographic algorithms such as **CRYSTALS-Kyber**.



- ❑Low latency.
- ❑Tightly-integrated read and write access to the CPU register file from the co-processor.
- ❑Unused opcodes within the CPU can be utilized.



Results



Two kinds of pseudo-instructions were implemented to integrate the butterfly unit into the NTT computation:

R-type (.insn r opcode6, func3, func7,rd, rs1, rs2)

R4-type (.insn r4 opcode6, func3, func2, rd, rs1, rs2, rs3)

The additional instructions are inserted using the "asm" keyword, which provides precise low-level hardware control, aiding performance optimization and interfacing with hardware. The method first tests various optimization flags for pure-sw implementation (**-Os**, **-O2**). Next, both type of instructions are evaluated.

Since each butterfly operation requires three 16-bit input data (*two coefficients, and the index of the twiddle factor*), a concatenation of the two input operands is necessary when using the R-type instruction.

Conversely, in the R4-type instructions with three source registers, this concatenation operation can be circumvented. The method also involves the exploitation of software optimization techniques like loop unrolling.

Références

[1] https://github.com/openhwgroup/core-v-xif/tree/main

[2] https://github.com/esl-epfl/cv32e40px

[3] Nannipieri P. et al. "A RISC-V Post Quantum Cryptography Instruction Set Extension for Number Theoretic Transform to Speed-Up CRYSTALS Algorithms". In: IEEE Access 9 (2021), pp. 150798–150808.